Java Chapter 2: Getting Started with Java

Contents

Time required: 90 minutes

Read: Think Java 2

One of the resources we will be using in this class is:

Think Java, Second Edition
by Allen Downey and Chris Mayfield

If you wish to read offline, the entire book is available in pdf: Think Java

Read the following interactive chapters. These have built in executable code snippets

- Chapter 1: Computer Programming
- Chapter 2: Variables and Operators

Do: Java Fundamentals Tutorials

- Java Tutorial
- Java Intro
- Java Get Started (Skip to Java Quickstart)

- Java Syntax
- Java Output
- Java Comments
- Java Variables
- Java Data Types
- Java User Input

Tutorial 1: Hello World

Hello World is the traditional first program you write in any programming language. Doing this confirms that your development environment is working.

Create a Java program named **HelloWorld.java**

```java
// Name: HelloWorld.java
// Written by:
// Written on:
// Purpose: Traditional Hello World in Java

public class HelloWorld {
    public static void main(String[] args) {
        // Print a string literal to the console
        System.out.println("Hello World!");
        System.out.println("This is my first Java Application");
    }
}
```

Example run:

```
Hello World!
This is my first Java Application
```

How Does This Work?

```java
// Traditional Hello World program in Java
```

Any line in a Java program starting with // is a comment. Comments are intended for others reading the code to understand the intent and functionality of the program. Comments are completely ignored by the Java compiler (the application that translates Java program to Java bytecode that computer can execute).

```
public class HelloWorld {
    ...
}
```

In Java, every application begins with a class definition. In the program, HelloWorld is the name of the class, and the class definition is between the curly braces {}

Every Java application has a class definition. The name of the class must match the filename in Java.

```
public static void main(String[] args) {
   ...
}
```

This is the main method. Every application in Java must contain the main method. The Java compiler starts executing the code from the main method.

The main function is the entry point of your Java application, and it's mandatory in a Java program.

```
System.out.println("Hello, World!");
```

The code above is a print statement. It prints the text Hello, World! to standard output (your screen). The text inside the quotation marks is called a String in Java.

Notice the print statement is inside the main function, which is inside the class definition.

Tutorial 2: Hello Dialog

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes can be used to display information or get input from the user.

A library is a collection of prewritten code to add capabilities to a program. **JOptionPane** is not included in the Java standard library. This is the syntax for importing a library in Java.

```
// Import JOptionPane library for Java GUI dialog boxes
import javax.swing.JOptionPane;
```

Create a Java program named **HelloDialog.java**

```java
// Name: HelloDialog.java
// Written by:
// Written on:
// Purpose: Traditional Hello World in Java JOptionPane

// Import JOptionPane library for Java GUI dialog boxes
import javax.swing.JOptionPane;
public class HelloDialog
{
   public static void main(String[] args)
   {
      // Show a message to the user
      JOptionPane.showMessageDialog(null, "Hello Java World!");
   }
}
```

Example run:

Tutorial 3: Miles to Kilometers

We are going to work with constants, variables, and expressions.

- **Constants:** store values that do not change
- **Variables:** store values that can change
- **Expression:** performs the work of the program

Constant: A constant is a variable whose value cannot change once it has been assigned. It is something we know before the program starts.

```java
// Constant for converting Miles to KM
final double KM_PER_MILE = 1.609;
```

Variables: A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type. A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.

```
// Variables for input and calculations
double miles = 100.0;
double kilometers;
```

Expressions: Expressions perform the work of a program. Expressions are used to compute and assign values to variables and to help control the execution flow of a program. The job of an expression is two-fold: perform the computation indicated by the elements of the expression and return some value.

```
// Convert miles to kilometers
kilometers = miles * KM_PER_MILE;
```

This program is hard coded to convert 100 miles to kilometers.

Create a Java program named: **Miles100ToKM.java**

```
1  // Name: Miles100ToKM.java
2  // Written by:
3  // Written on:
4  // Purpose: Convert 100 miles to Kilometers
5
6  public class Miles100ToKM {
7
8      public static void main(String[] args) {
9          // Constant for converting Miles to KM
10         final double KM_PER_MILE = 1.609;
11
12         // Variables for input and calculations
13         double miles = 100.0;
14         double kilometers;
15
16         // Convert miles to kilometers
17         kilometers = miles * KM_PER_MILE;
18
19         // Display answer to user
20         System.out.println(miles + " miles = " + kilometers + " kilometers");
21     }
22 }
```

Example run:

```
100.0 miles = 160.9 kilometers
```

Tutorial 3: Get Input

Scanner: It is used to get input from the user during runtime. The input is to be of primitive data types, like int, float, double, string, etc.

In order to use the object of Scanner, we need to import **java.util.Scanner** package.

```
import java.util.Scanner
```

We create an object of the `Scanner` class. We can use the object to take input from the user.

```
// Create an object of Scanner
Scanner keyboard = new Scanner(System.in);

// Get input from the user
int number = keyboard.nextInt();
```

Create a Java program named **GetInput.java**

```java
// Name: GetInput.java
// Written by:
// Written on:
// Purpose: Demonstrate methods of input

// Import Scanner library for input
import java.util.Scanner;

class GetInput {
    public static void main(String[] args) {
        // Create scanner object
        Scanner keyboard = new Scanner(System.in);

        // Get int input
        System.out.print("Enter int: ");
        int myInt = keyboard.nextInt();
        System.out.println("Int entered = " + myInt);

        // Get double input
        System.out.print("Enter double: ");
        double myDouble = keyboard.nextDouble();
        System.out.println("Double entered = " + myDouble);

        // Get String input
        System.out.print("Enter text: ");
        String myString = keyboard.next();
        System.out.println("Text entered = " + myString);
        keyboard.close();
    }
}
```

Example run:

```
Enter int: 256
Int entered = 256
Enter double: 3.145
Double entered = 3.145
Enter text: Hello!
Text entered = Hello!
```

Tutorial 4: Miles to Kilometers with User Input

The same program we did earlier with user input using the Scanner class.

Create a Java program named: **MilesToKM.java**

```java
// Name: MilesToKM.java
// Written by:
// Written on:
// Purpose: Convert Miles to Kilometers from user input

// Import Scanner library for input
import java.util.Scanner;

public class MilesToKM {
    public static void main(String[] args) {
        // Declare Scanner object and initialize with
        // predefined standard input object, System.in
        Scanner keyboard = new Scanner(System.in);

        // Constant for converting Miles to KM
        final double KM_PER_MILE = 1.609;

        // Variables for input and calculations
        double miles;
        double kilometers;

        // Prompt the user for input
        System.out.println("+-----------------------------------+");
        System.out.println("|--  Convert Miles to Kilometers  --|");
        System.out.println("+-----------------------------------+");
        System.out.print("Enter the distance in miles: ");

        // Get double from the keyboard
        // Assign double to variable
        miles = keyboard.nextDouble();

        // Convert miles to kilometers
        kilometers = miles * KM_PER_MILE;

        // Display answer to user
        System.out.println(miles + " miles = " + kilometers + " kilometers");
        // Close scanner object
        keyboard.close();
    }
}
```

Example run:

```
+-----------------------------------+
|--  Convert Miles to Kilometers  --|
+-----------------------------------+
Enter the distance in miles: 10
10.0 miles = 16.09 kilometers
```

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.